FIG. 1

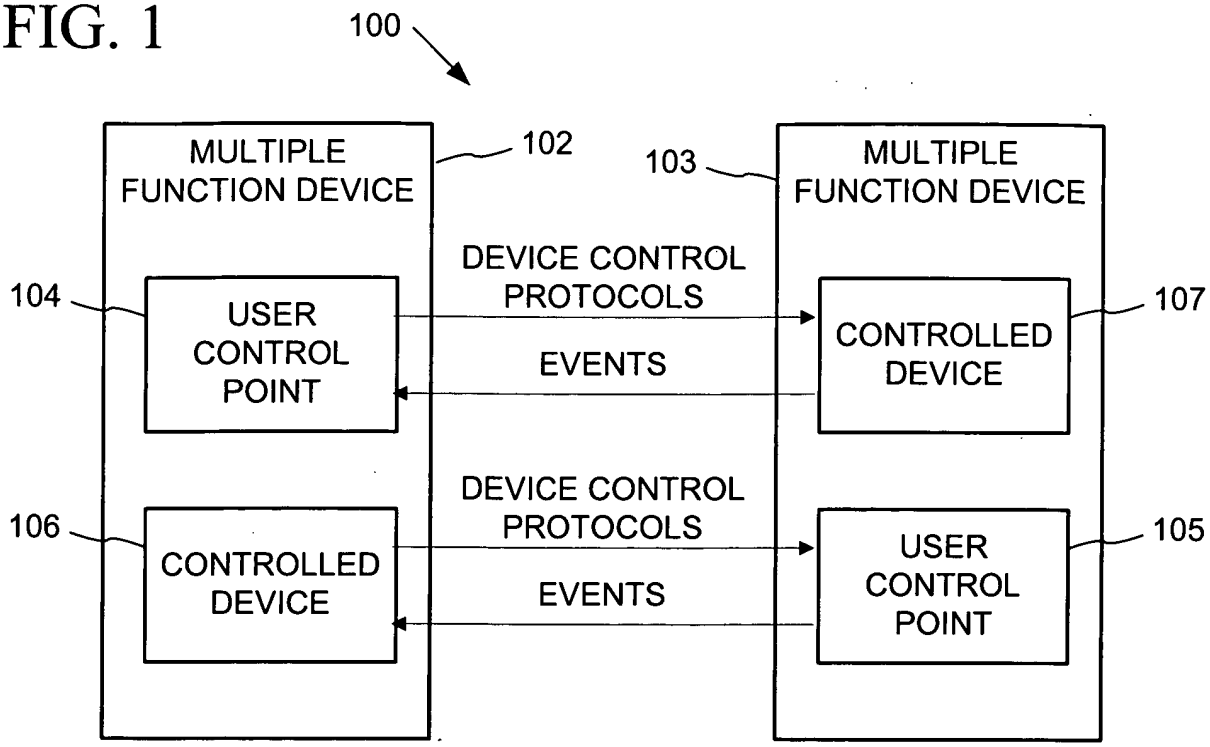
100
MULTIPLE FUNCTION DEVICE
102
103
MULTIPLE FUNCTION DEVICE
DEVICE CONTROL PROTOCOLS
104
USER CONTROL POINT
EVENTS
CONTROLLED DEVICE
107
DEVICE CONTROL PROTOCOLS
106
CONTROLLED DEVICE
EVENTS
USER CONTROL POINT
105

FIG. 2

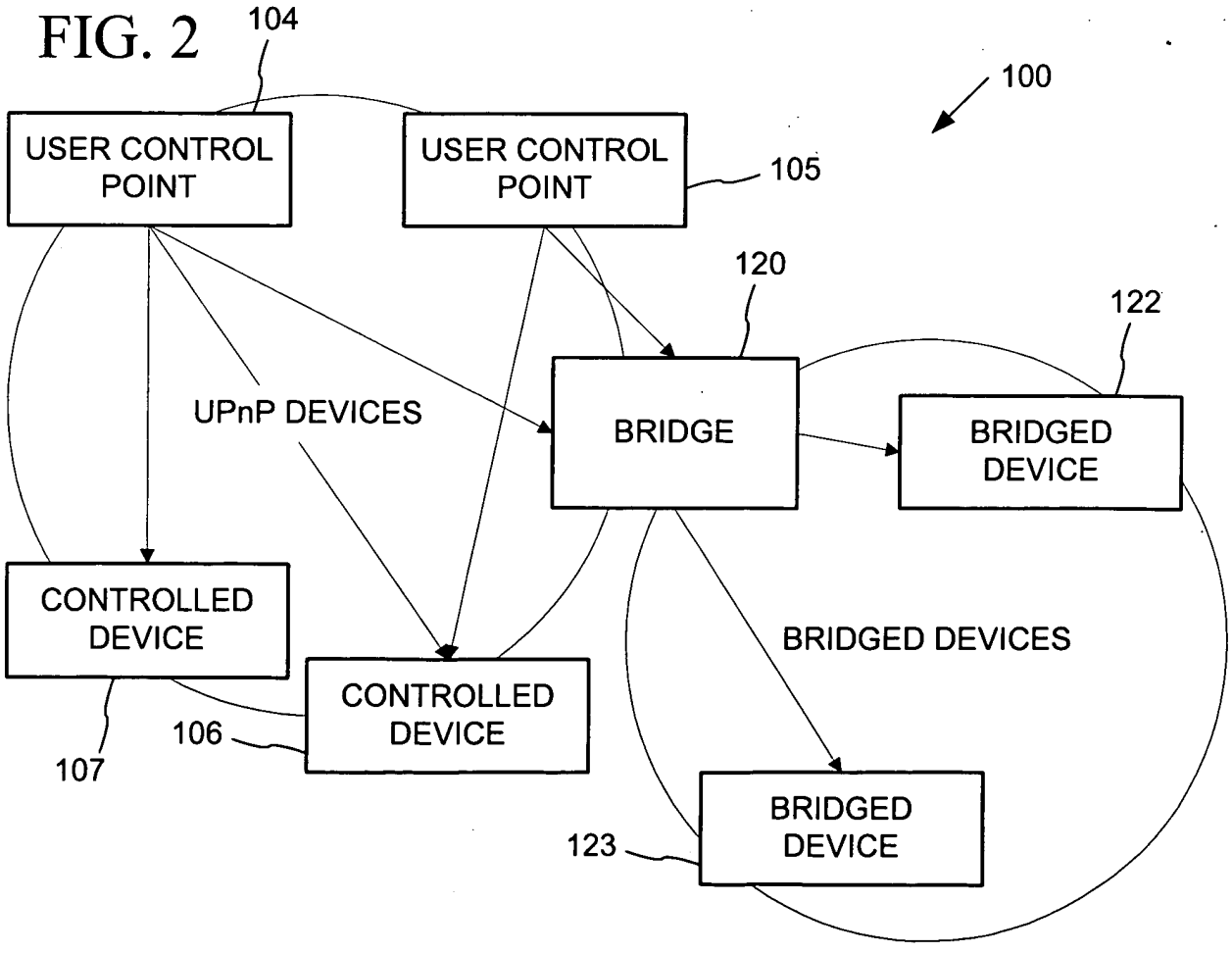
104
100
USER CONTROL POINT
USER CONTROL POINT
105
120
122
UPnP DEVICES
BRIDGE
BRIDGED DEVICE
CONTROLLED DEVICE
BRIDGED DEVICES
CONTROLLED DEVICE
106
107
BRIDGED DEVICE
123

FIG. 3

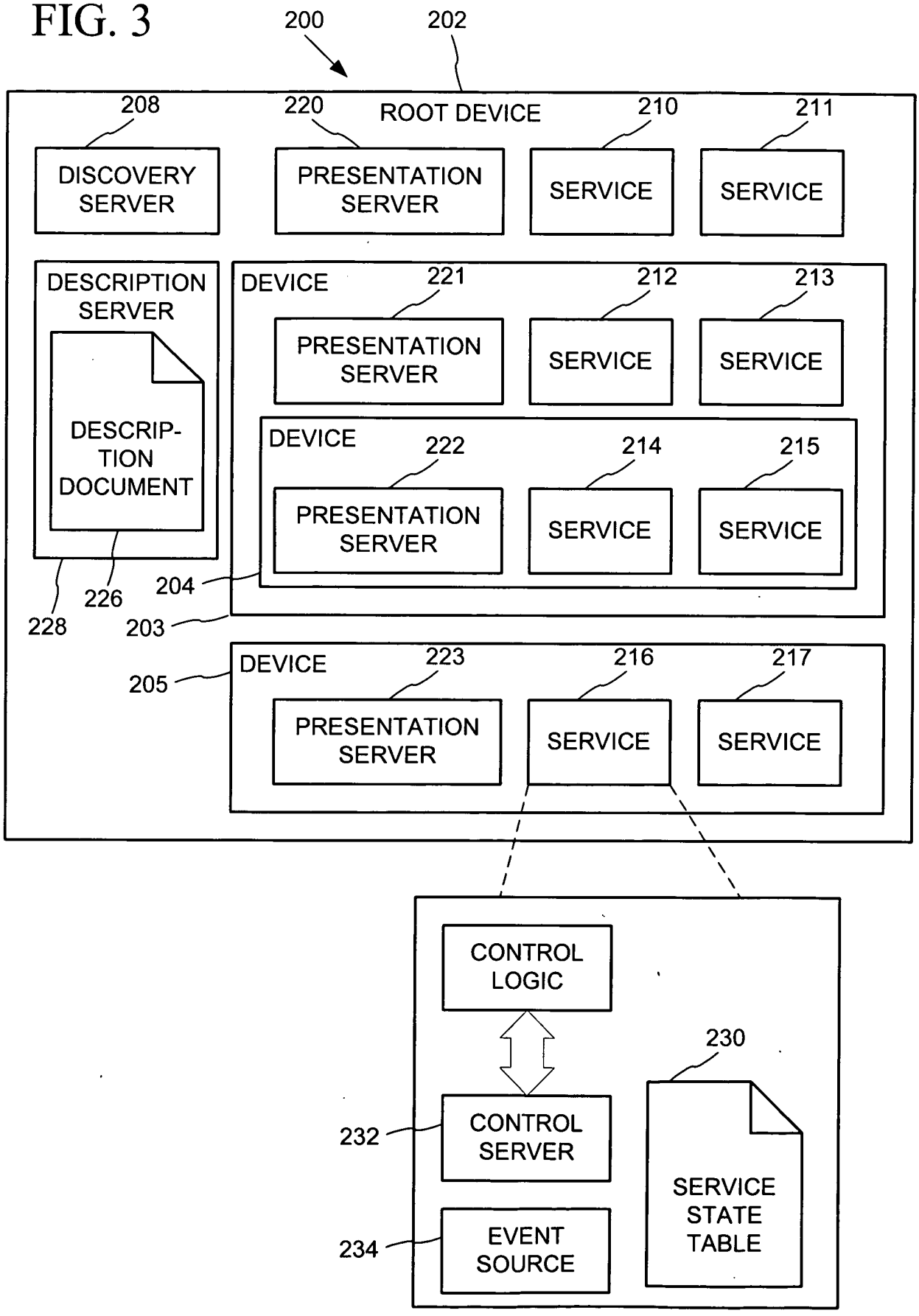
200
202
208
220
ROOT DEVICE
210
211
DISCOVERY SERVER
PRESENTATION SERVER
SERVICE
SERVICE
DESCRIPTION SERVER
DESCRIP-TION DOCUMENT
DEVICE
221
212
213
PRESENTATION SERVER
SERVICE
SERVICE
DEVICE
222
214
215
PRESENTATION SERVER
SERVICE
SERVICE
204
226
228
203
205
DEVICE
223
216
217
PRESENTATION SERVER
SERVICE
SERVICE
CONTROL LOGIC
230
232
CONTROL SERVER
SERVICE STATE TABLE
234
EVENT SOURCE

FIG. 4

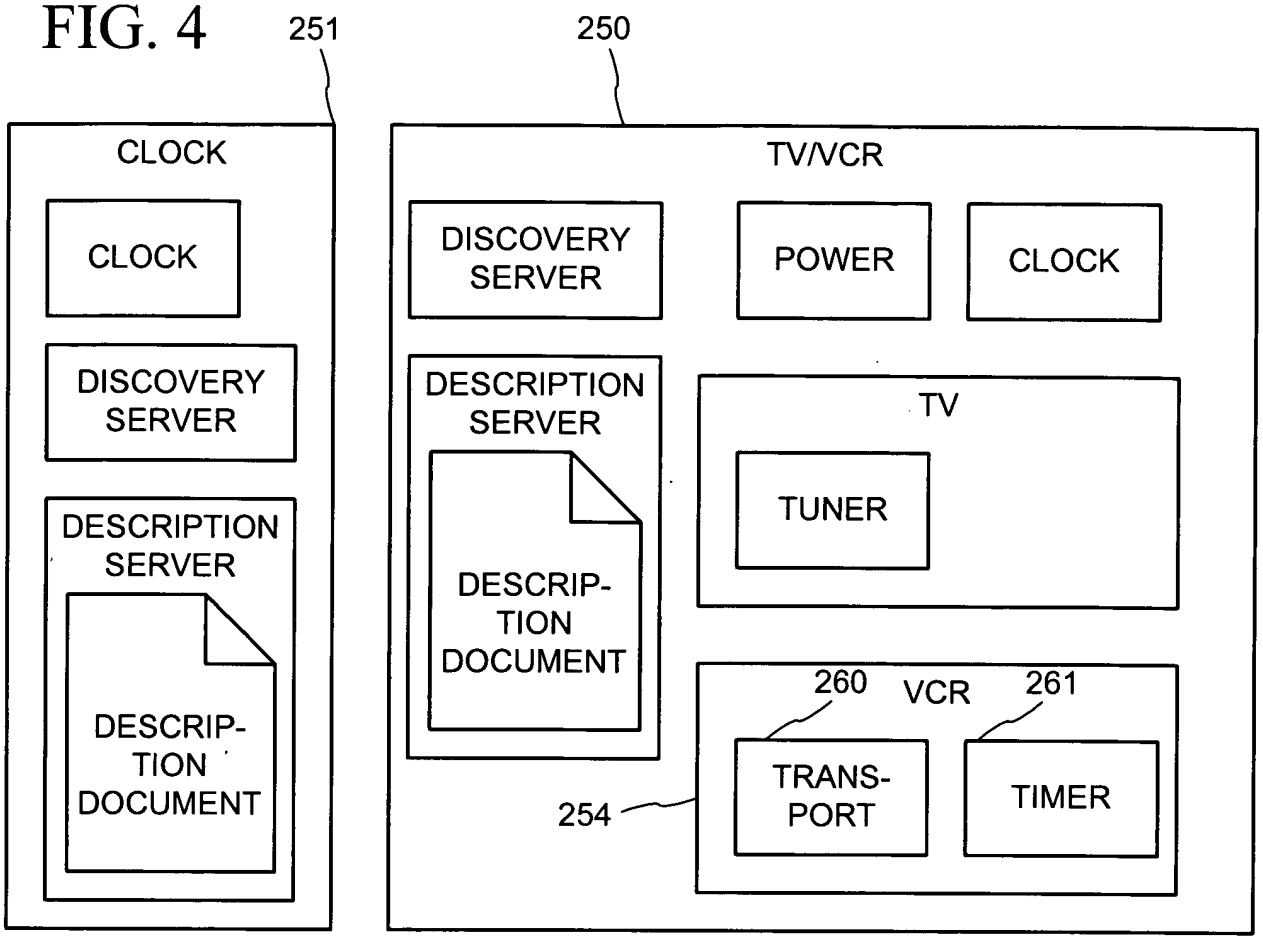
251
250
CLOCK
CLOCK
DISCOVERY SERVER
DESCRIPTION SERVER
DESCRIP-TION DOCUMENT
TV/VCR
DISCOVERY SERVER
POWER
CLOCK
DESCRIPTION SERVER
DESCRIP-TION DOCUMENT
TV
TUNER
260
VCR
261
TRANS-PORT
TIMER
254

FIG. 5

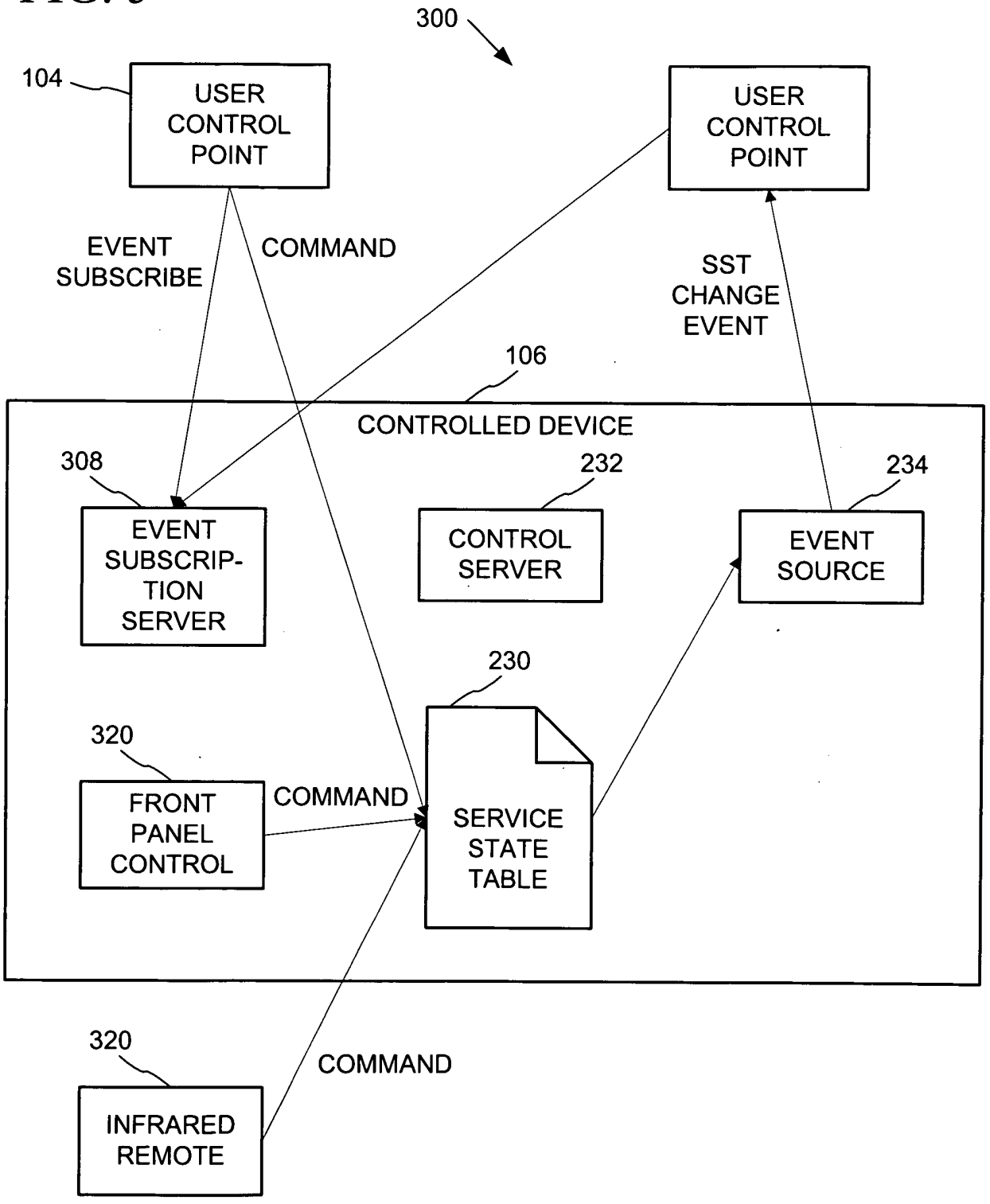

FIG. 6

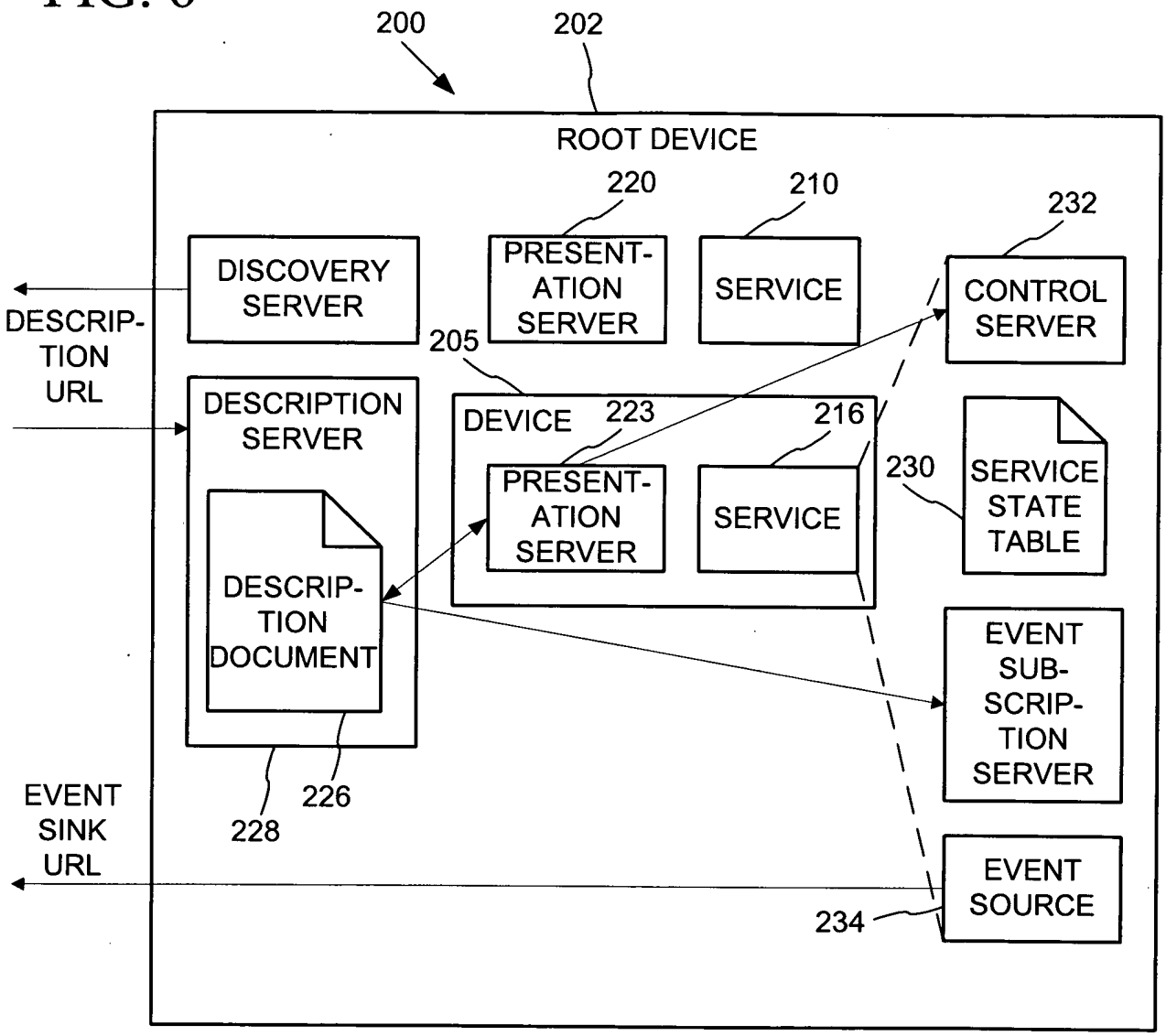
200
202
ROOT DEVICE
220
210
232
DISCOVERY SERVER
PRESENT-ATION SERVER
SERVICE
CONTROL SERVER
DESCRIP-TION URL
205
DESCRIPTION SERVER
DEVICE
223
216
PRESENT-ATION SERVER
SERVICE
230
SERVICE STATE TABLE
DESCRIP-TION DOCUMENT
EVENT SUB-SCRIP-TION SERVER
EVENT SINK URL
228
226
EVENT SOURCE
234

FIG. 7

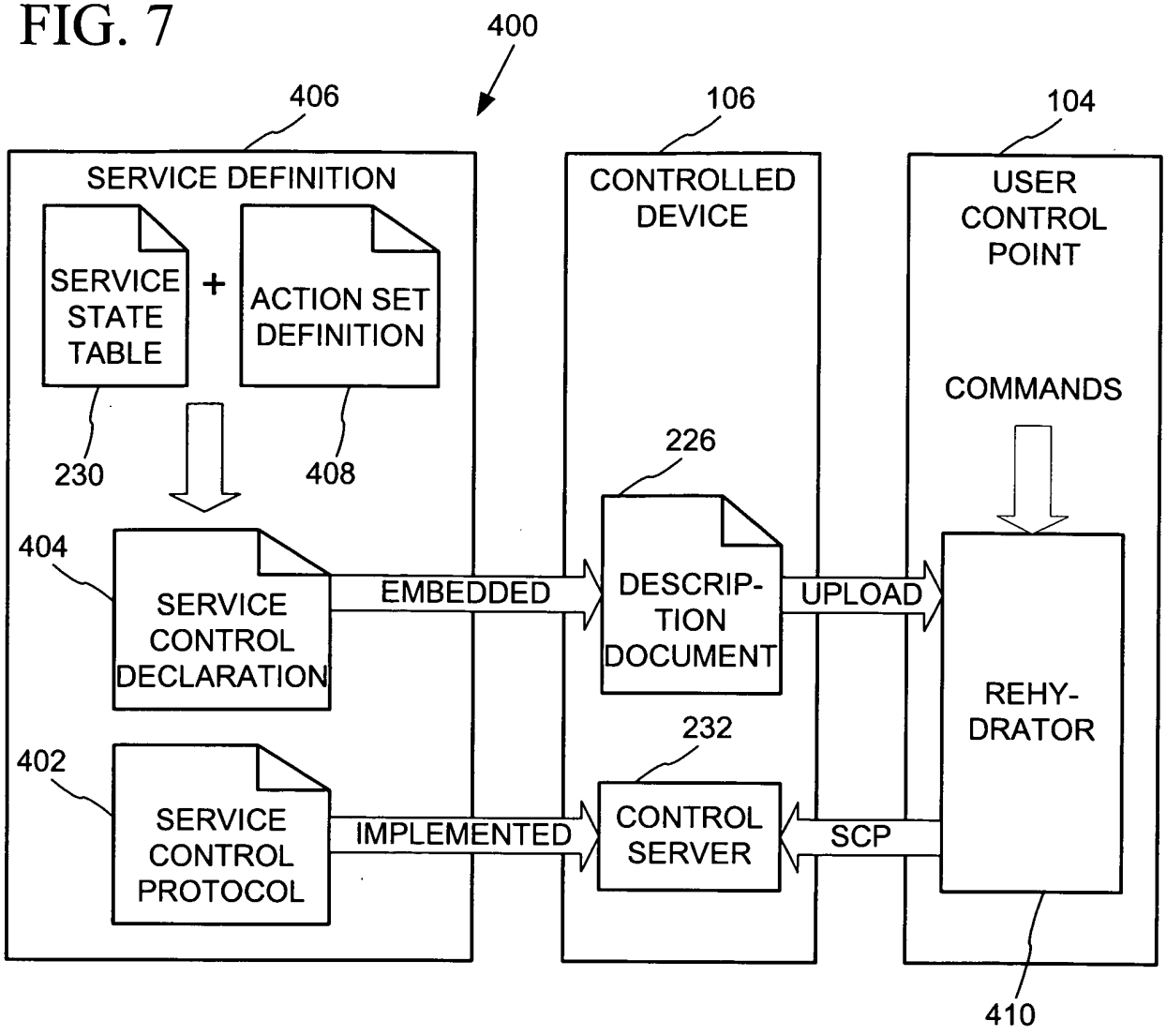

FIG. 8

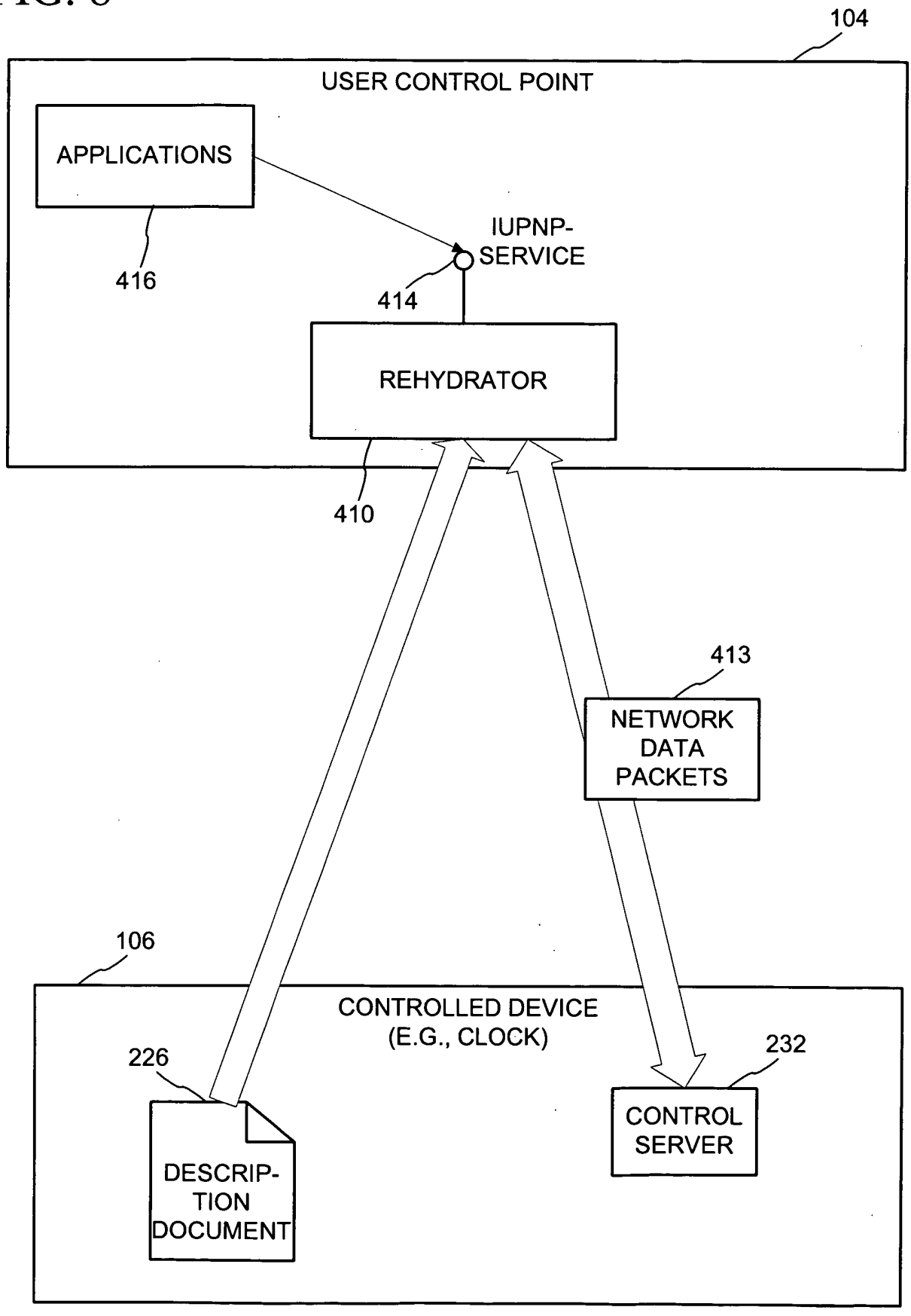
104
USER CONTROL POINT
APPLICATIONS
416
IUPNP-
SERVICE
414
REHYDRATOR
410
413
NETWORK
DATA
PACKETS
106
CONTROLLED DEVICE
(E.G., CLOCK)
226
DESCRIP-
TION
DOCUMENT
232
CONTROL
SERVER

FIG. 9

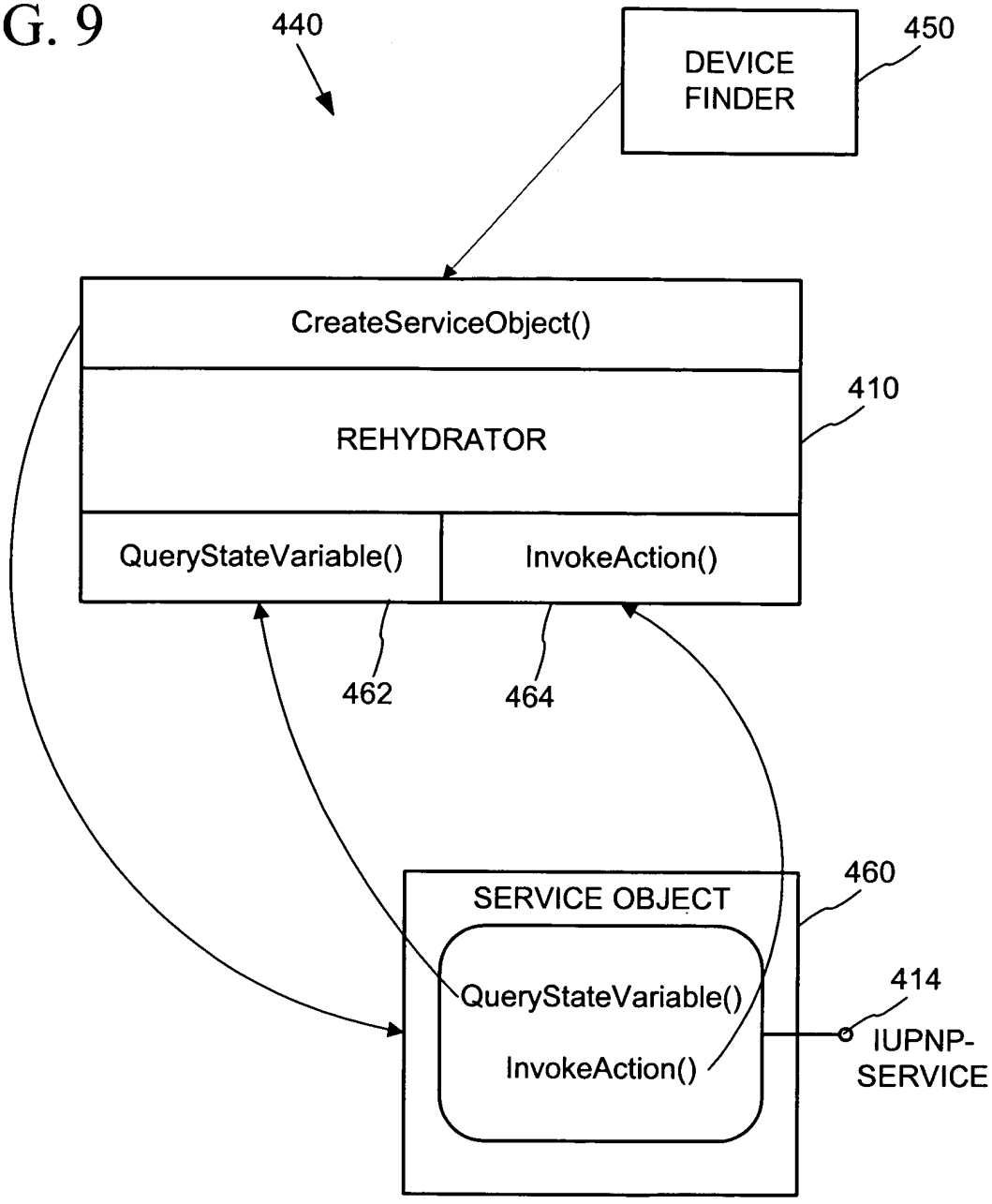
440
DEVICE FINDER
450
CreateServiceObject()
REHYDRATOR
410
QueryStateVariable()
InvokeAction()
462
464
SERVICE OBJECT
460
QueryStateVariable()
InvokeAction()
414
IUPNP-SERVICE

FIG. 10

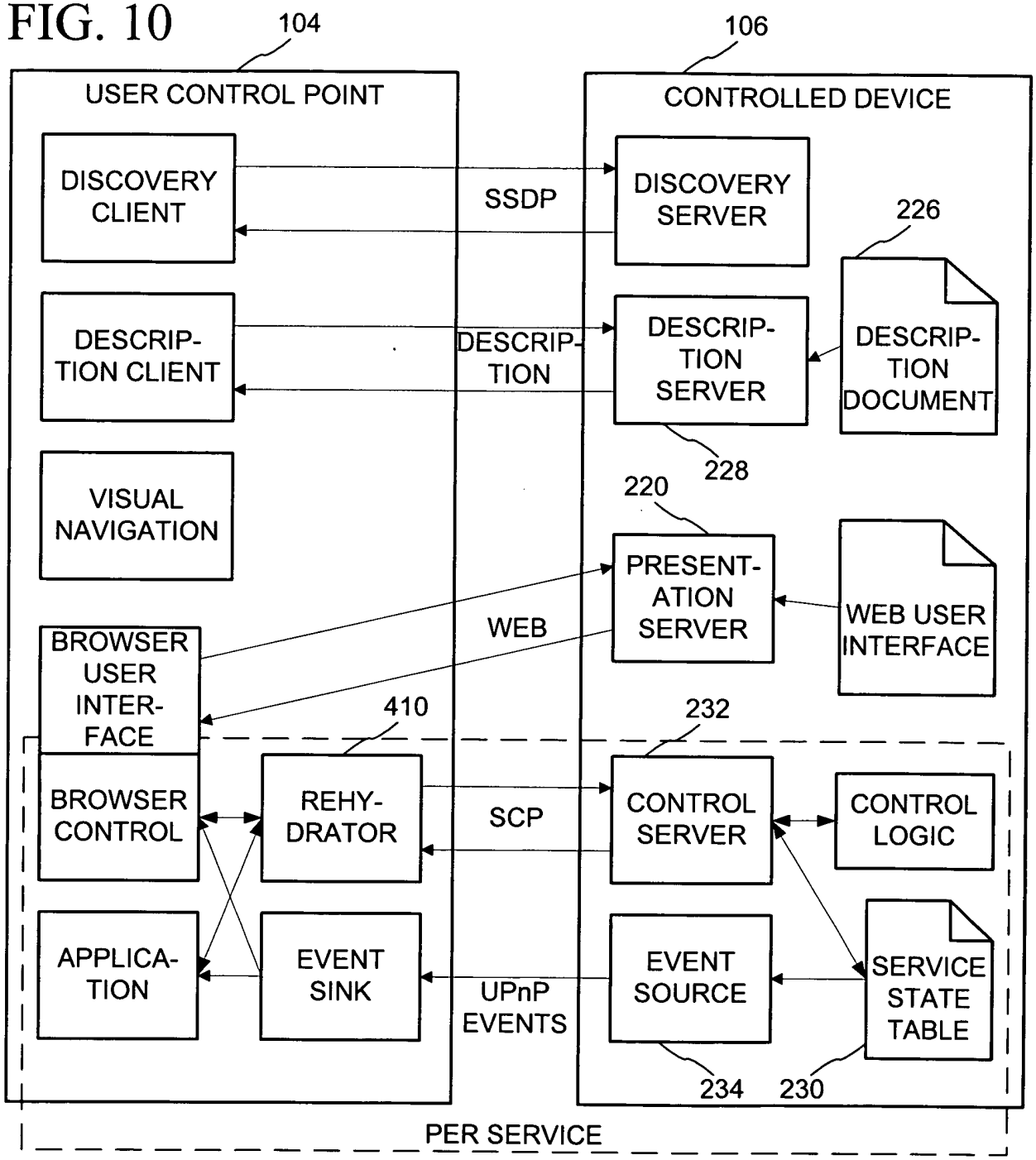
104
106
USER CONTROL POINT
CONTROLLED DEVICE
DISCOVERY CLIENT
SSDP
DISCOVERY SERVER
226
DESCRIP-TION CLIENT
DESCRIP-TION
DESCRIP-TION SERVER
DESCRIP-TION DOCUMENT
228
VISUAL NAVIGATION
220
PRESENT-ATION SERVER
WEB USER INTERFACE
WEB
BROWSER USER INTER-FACE
410
232
BROWSER CONTROL
REHY-DRATOR
SCP
CONTROL SERVER
CONTROL LOGIC
APPLICA-TION
EVENT SINK
UPnP EVENTS
EVENT SOURCE
SERVICE STATE TABLE
234
230
PER SERVICE

FIG. 11

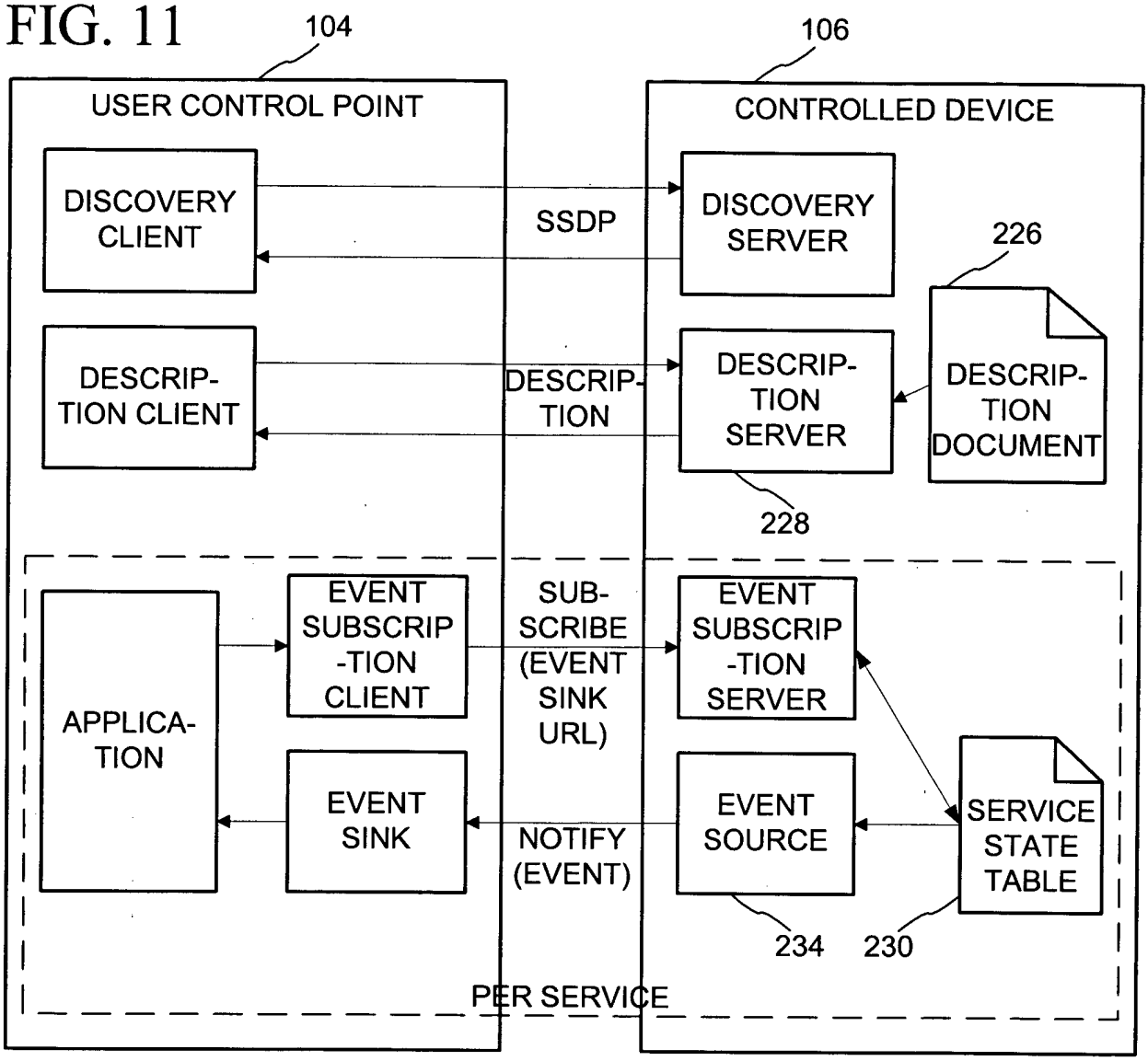
104
106
USER CONTROL POINT
CONTROLLED DEVICE
DISCOVERY CLIENT
SSDP
DISCOVERY SERVER
226
DESCRIP-TION CLIENT
DESCRIP-TION
DESCRIP-TION SERVER
DESCRIP-TION DOCUMENT
228
APPLICA-TION
EVENT SUBSCRIP-TION CLIENT
SUB-SCRIBE (EVENT SINK URL)
EVENT SUBSCRIP-TION SERVER
EVENT SINK
NOTIFY (EVENT)
EVENT SOURCE
SERVICE STATE TABLE
234
230
PER SERVICE

FIG. 12

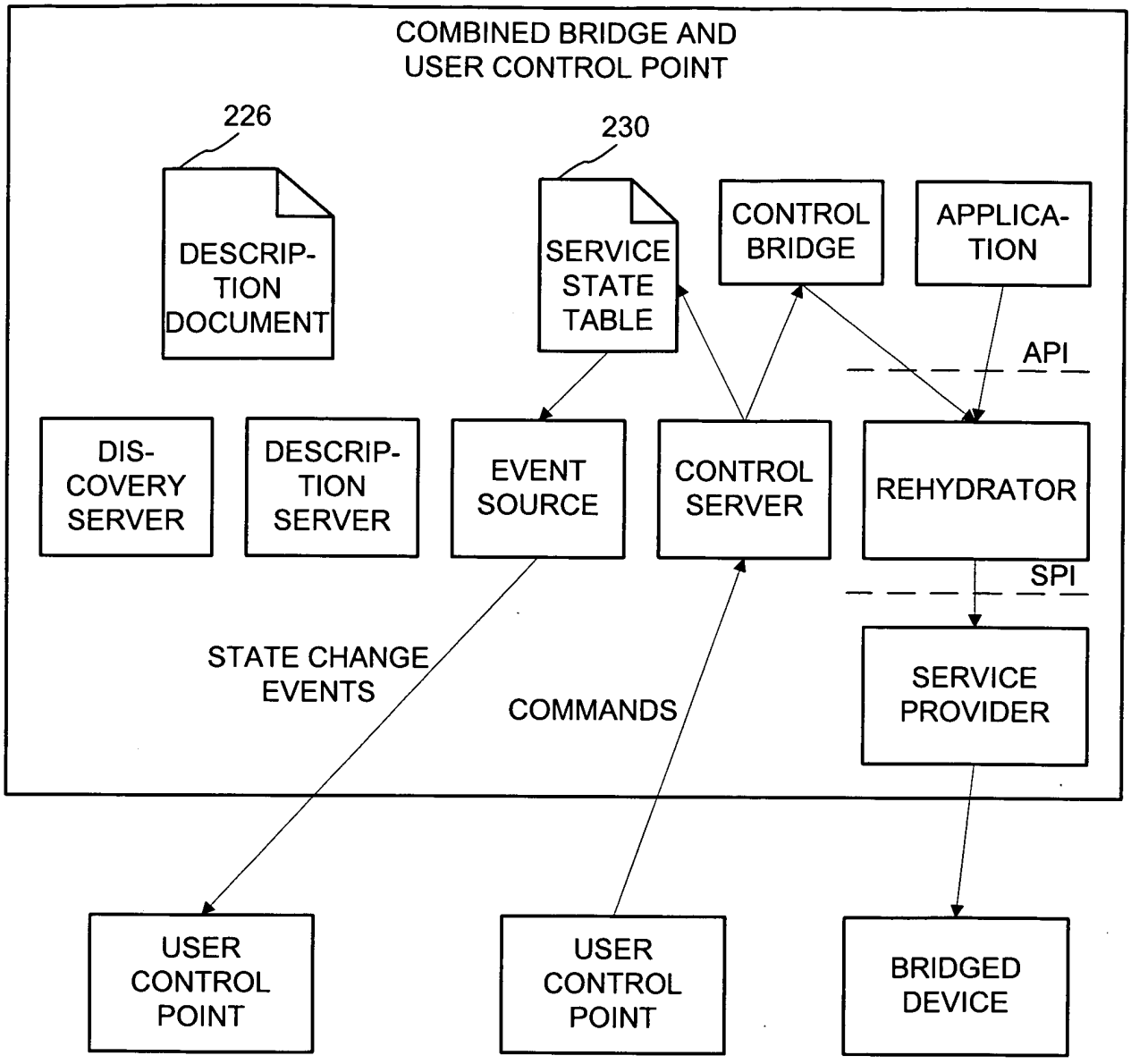

FIG. 13

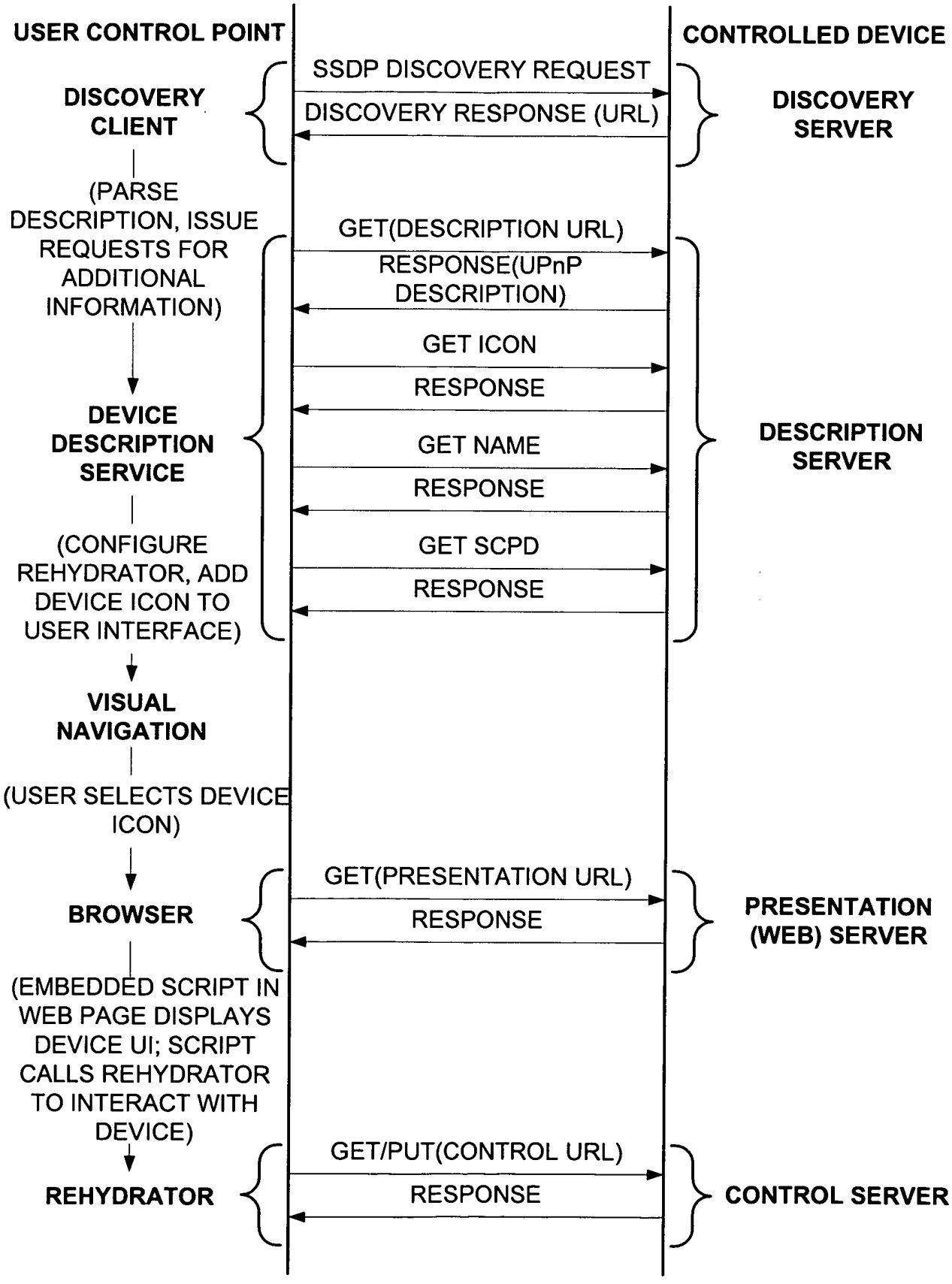

FIG. 14

```
<?xml version="1.0"?>
<root xmlns="urn:schemas-upnp-org:device-1-0">
 <specVersion>
  <major>1</major>
  <minor>0</minor>
 </specVersion>
 <URLBase>base URL for all relative URLs</URLBase>
 <device>
  <deviceType>urn:schemas-upnp-org:device:deviceType:v</deviceType>
  <friendlyName>short user-friendly title</friendlyName>
  <manufacturer>manufacturer name</manufacturer>
  <manufacturerURL>URL to manufacturer site</manufacturerURL>
  <modelDescription>long user-friendly title</modelDescription>
  <modelName>model name</modelName>
  <modelNumber>model number</modelNumber>
  <modelURL>URL to model site</modelURL>
  <serialNumber>manufacturer's serial number</serialNumber>
  <UDN>uuid:UUID</UDN>
  <UPC>Universal Product Code</UPC>
  <iconList>
   <icon>
    <mimetype>image/format</mimetype>
    <width>horizontal pixels</width>
    <height>vertical pixels</height>
    <depth>color depth</depth>
    <url>URL to icon</url>
   </icon>
   XML to declare other icons, if any, go here
  </iconList>
  <serviceList>
   <service>
    <serviceType>urn:schemas-upnp-org:service:serviceType:v</serviceType>
    <serviceId>urn:upnp-org:serviceId:serviceID</serviceId>
    <SCPDURL>URL to service description</SCPDURL>
    <controlURL>URL for control</controlURL>
    <eventSubURL>URL for eventing</eventSubURL>
   </service>
   Declarations for other services (if any) go here
  </serviceList>
  <deviceList>
   Description of embedded devices (if any) go here
  </deviceList>
  <presentationURL>URL for presentation</presentationURL>
 </device>
</root>
```

FIG. 15

```
<device>
  ...
  <iconList>
    <icon>
      <size>16</size>
      <color>0</color>
      <depth>8</depth>
      <imageType>PNG</imageType>
      <image>"http://device.local/iconpath/icon16bw.png"</image>
    </icon>
    <icon>
      <size>32</size>
      <color>0</color>
      <depth>8</depth>
      <imageType>PNG</imageType>
      <image>"http://device.local/iconpath/icon32bw.png"</image>
    </icon>
    <icon>
      <size>48</size>
      <color>0</color>
      <depth>8</depth>
      <imageType>PNG</imageType>
      <image>"http://device.local/iconpath/icon48bw.png"</image>
    </icon>
    <icon>
      <size>16</size>
      <color>1</color>
      <depth>8</depth>
      <imageType>PNG</imageType>
      <image>"http://device.local/iconpath/icon16c.png"</image>
    </icon><device>
    <icon>
      <size>32</size>
      <color>0</color>
      <depth>8</depth>
      <imageType>PNG</imageType>
      <image>"http://device.local/iconpath/icon32c.png"</image>
    </icon>
    <icon>
      <size>48</size>
      <color>0</color>
      <depth>8</depth>
      <imageType>PNG</imageType>
      <image>"http://device.local/iconpath/icon48c.png"</image>
    </icon>
    ...
  </iconList>
  ...
</device>
```

FIG. 16

```
<?xml version="1.0"?>
<scpd xmlns="urn:schemas-upnp-org:service-1-0">
 <specVersion>
  <major>1</major>
  <minor>0</minor>
 </specVersion>
 <actionList>
  <action>
   <name>actionName</name>
   <argumentList>
    <argument>
     <name>formalParameterName</name>
     <direction>in xor out</direction>
     <retval />
     <relatedStateVariable>stateVariableName</relatedStateVariable>
    </argument>
    Declarations for other arguments (if any) go here
   </argumentList>
  </action>
  Declarations for other actions (if any) go here
 </actionList>
 <serviceStateTable>
  <stateVariable sendEvents="yes">
   <name>variableName</name>
   <dataType>variable data type</dataType>
   <defaultValue>default value</defaultValue>
   <allowedValueList>
    <allowedValue>enumerated value</allowedValue>
    Other allowed values (if any) go here
   </allowedValueList>
  </stateVariable>
  <stateVariable sendEvents="yes">
   <name>variableName</name>
   <dataType>variable data type</dataType>
   <defaultValue>default value</defaultValue>
   <allowedValueRange>
    <minimum>minimum value</minimum>
    <maximum>maximum value</maximum>
    <step>increment value</step>
   </allowedValueRange>
  </stateVariable>
  Declarations for other state variables (if any) go here
 </serviceStateTable>
</scpd>
```

FIG. 17

```
<?xml version="1.0"?>
<Schema name="upnp_scpdl"
   xmlns="urn:schemas-microsoft-com:xml-data"
   xmlns:dt="urn:schemas-microsoft-com:datatypes">

   <!-- Common Elements and Attributes -->

   <ElementType name="name" content="textOnly" dt:type="string" />

   <!-- Service State Table -->

   <ElementType name="minimum" content="textOnly" dt:type="number" />
   <ElementType name="maximum" content="textOnly" dt:type="number" />
   <ElementType name="step" content="textOnly" dt:type="number" />

   <ElementType name="allowedValueRange" content="eltOnly" model="closed">
      <element type="minimum" />
      <element type="maximum" />
      <element type="step" minOccurs="0" />
   </ElementType>

   <ElementType name="allowedValue" content="textOnly" />

   <ElementType name="allowedValueList" content="eltOnly" model="closed">
      <element type="allowedValue" minOccurs="1" maxOccurs="*" />
   </ElementType>

   <ElementType name="dataType" content="textOnly" dt:type="string" />

   <ElementType name="stateVariable" content="eltOnly" model="closed">
      <element type="name" />
...
```

FIG. 18

```
...
      <element type="dataType" />

      <group minOccurs="0" maxOccurs="1" order="one">
        <element type="allowedValueRange" />
        <element type="allowedValueList" />
      </group>
    </ElementType>


    <ElementType name="deviceStateTable" content="eltOnly" model="closed">
      <element type="stateVariable" minOccurs="1" maxOccurs="*" />
    </ElementType>


    <!-- Action List -->

    <ElementType name="relatedStateVariable" content="textOnly" dt:type="string" />

    <ElementType name="argument" content="eltOnly" model="closed">
      <element type="name" />
      <element type="relatedStateVariable" />
    </ElementType>

    <ElementType name="action" content="eltOnly" model="closed">
      <element type="name" />
      <element type="argument" minOccurs="0" maxOccurs="*" />
    </ElementType>

    <ElementType name="actionList" content="eltOnly" model="closed">
      <element type="action" minOccurs="0" maxOccurs="*" />
    </ElementType>

    <!-- Root Element -->

    <ElementType name="dcpd" content="eltOnly" model="closed">
      <element type="deviceStateTable" />
      <element type="actionList" />
    </ElementType>
  </Schema>
```

FIG. 19

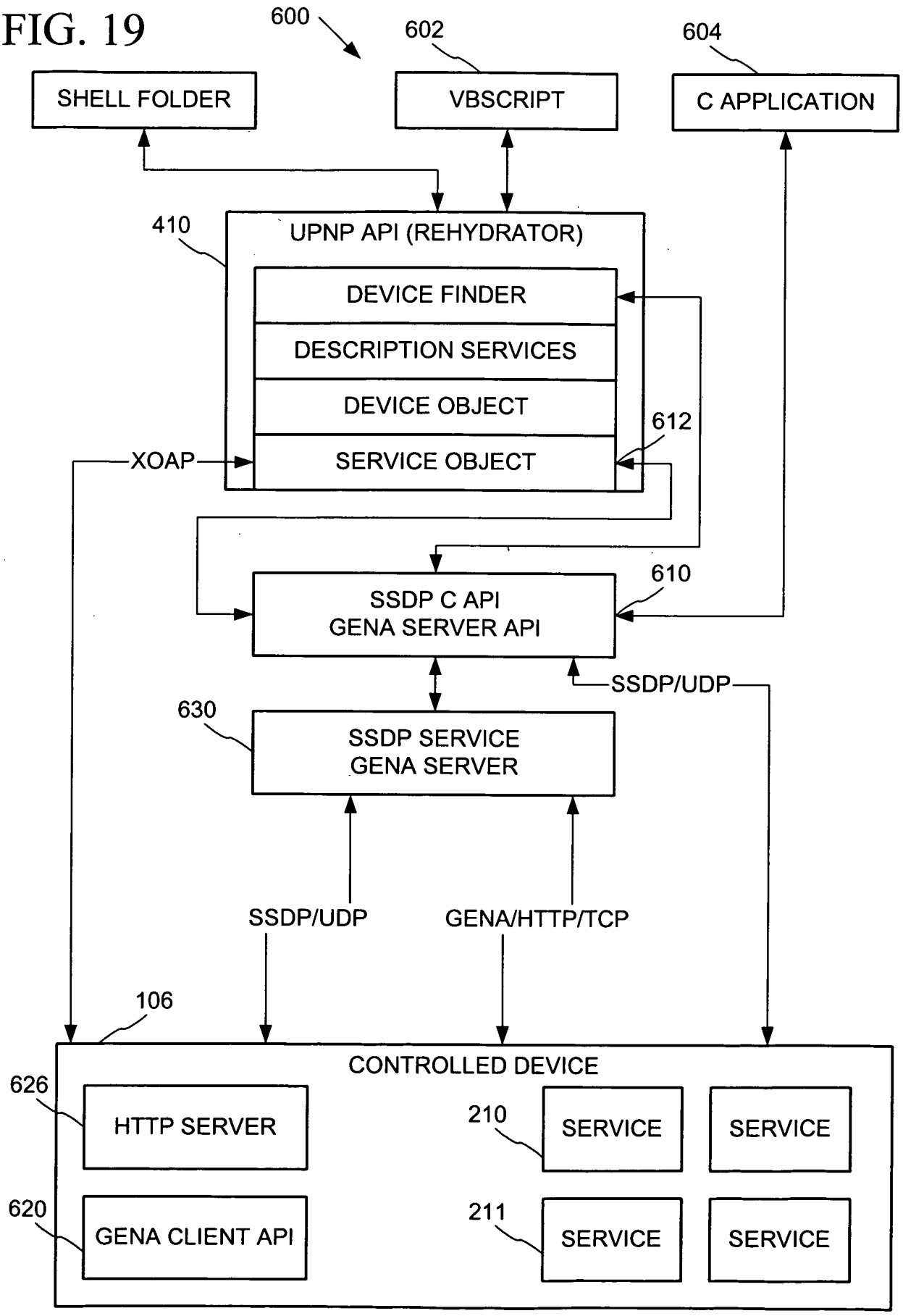
600
602
604
SHELL FOLDER
VBSCRIPT
C APPLICATION
410
UPNP API (REHYDRATOR)
DEVICE FINDER
DESCRIPTION SERVICES
DEVICE OBJECT
SERVICE OBJECT
612
XOAP
610
SSDP C API
GENA SERVER API
SSDP/UDP
630
SSDP SERVICE
GENA SERVER
SSDP/UDP
GENA/HTTP/TCP
106
CONTROLLED DEVICE
626
HTTP SERVER
210
SERVICE
SERVICE
620
GENA CLIENT API
211
SERVICE
SERVICE

FIG. 20

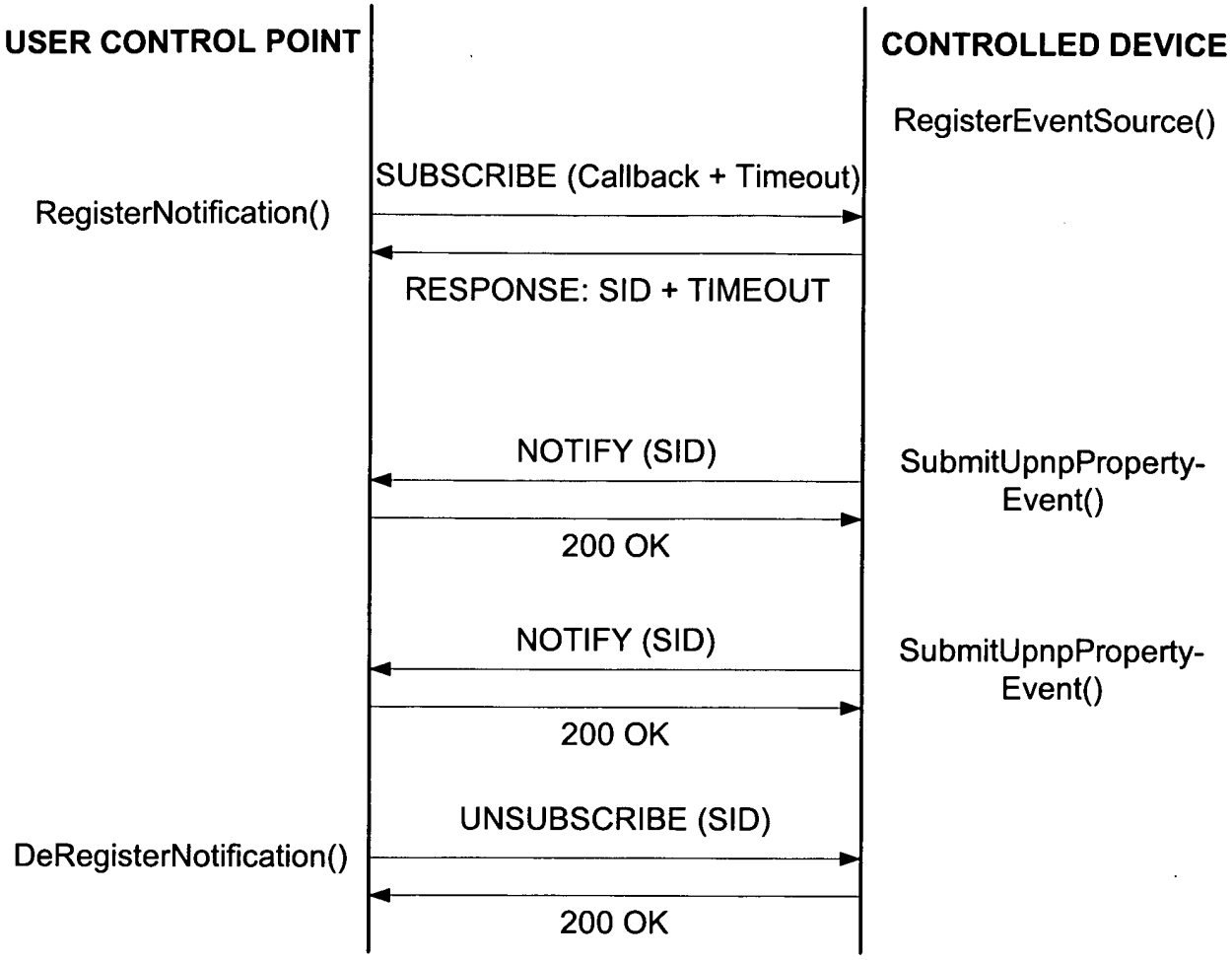

FIG. 21

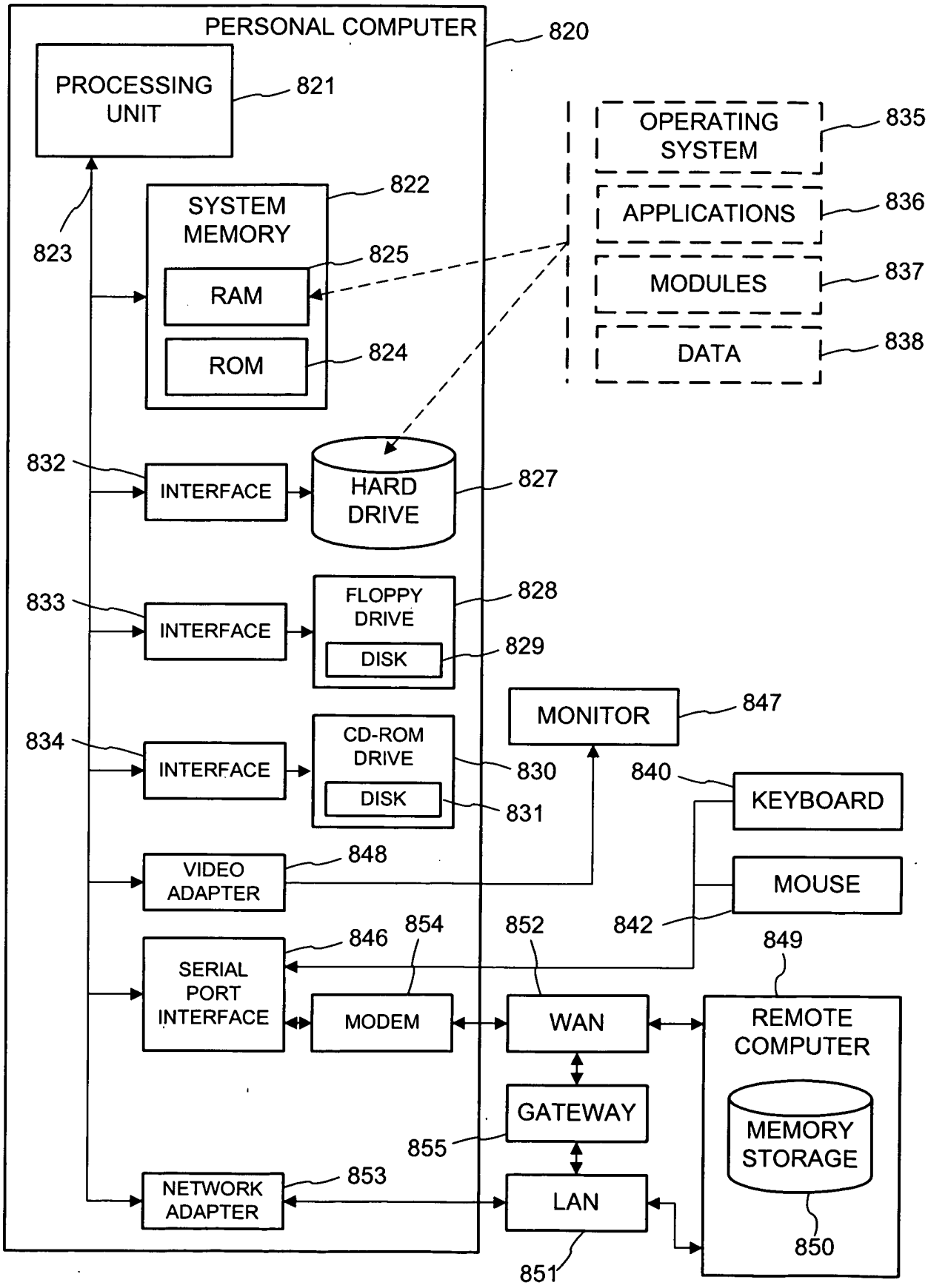
PERSONAL COMPUTER
820
PROCESSING UNIT
821
SYSTEM MEMORY
822
RAM
825
ROM
824
823
OPERATING SYSTEM
835
APPLICATIONS
836
MODULES
837
DATA
838
832
INTERFACE
HARD DRIVE
827
833
INTERFACE
FLOPPY DRIVE
828
DISK
829
834
INTERFACE
CD-ROM DRIVE
830
DISK
831
MONITOR
847
840
KEYBOARD
VIDEO ADAPTER
848
MOUSE
842
846
854
852
849
SERIAL PORT INTERFACE
MODEM
WAN
REMOTE COMPUTER
GATEWAY
855
MEMORY STORAGE
NETWORK ADAPTER
853
LAN
850
851

FIG. 22

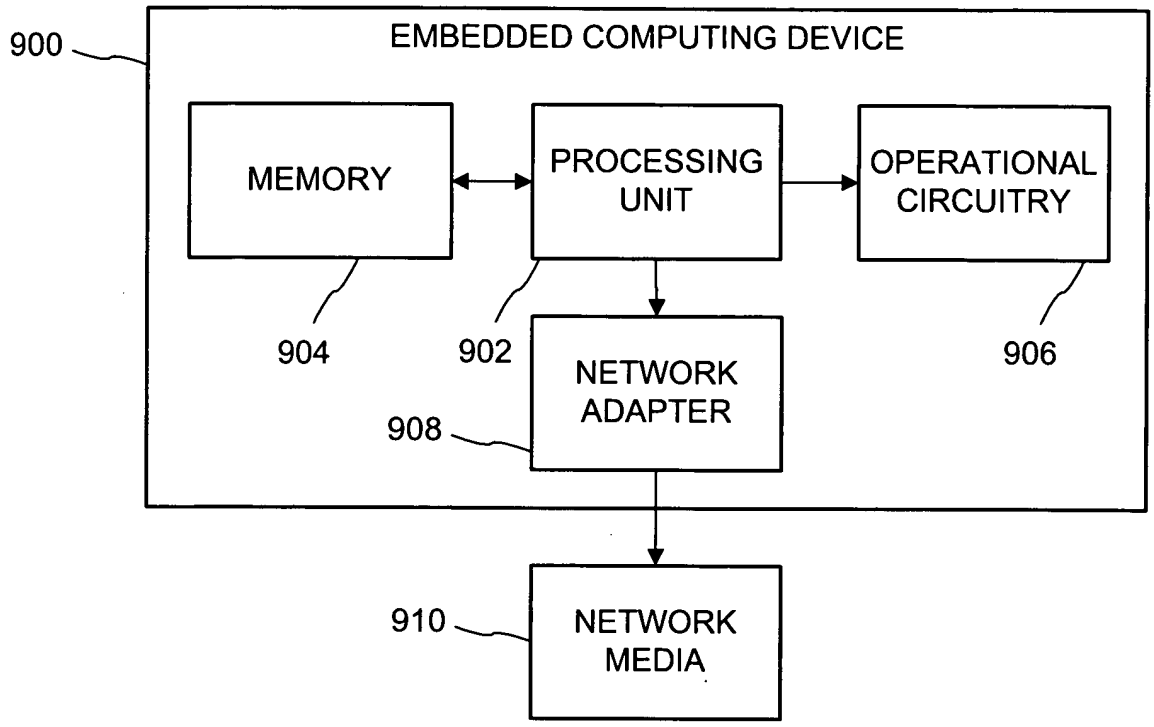
900
EMBEDDED COMPUTING DEVICE
MEMORY
PROCESSING UNIT
OPERATIONAL CIRCUITRY
904
902
906
NETWORK ADAPTER
908
910
NETWORK MEDIA

FIG. 23

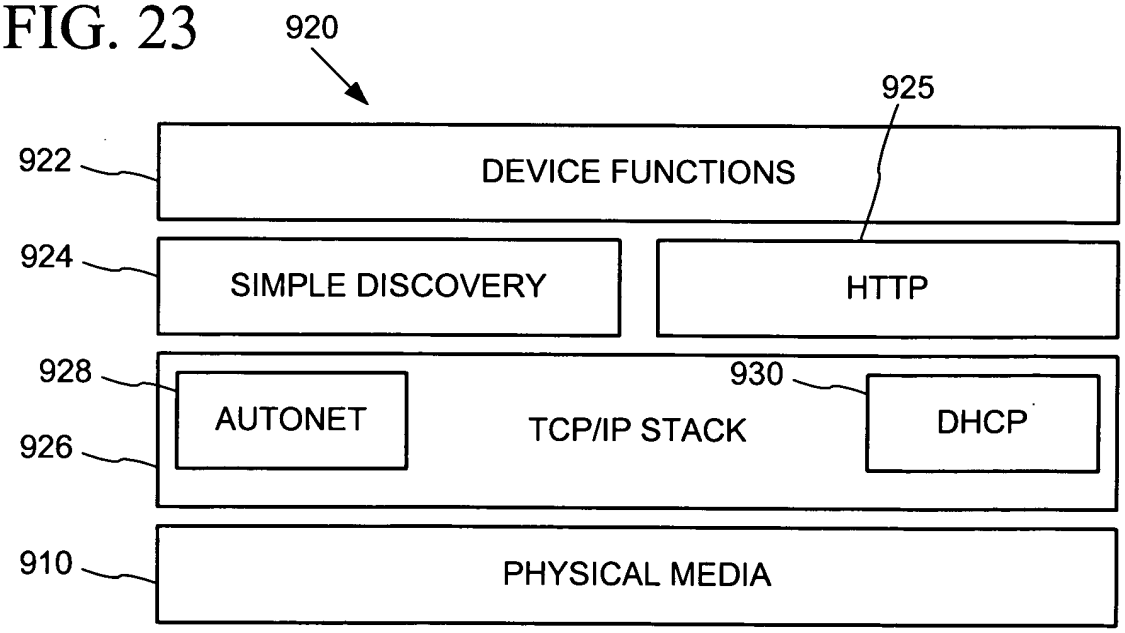
920
925
922
DEVICE FUNCTIONS
924
SIMPLE DISCOVERY
HTTP
928
930
AUTONET
TCP/IP STACK
DHCP
926
910
PHYSICAL MEDIA

FIG. 24

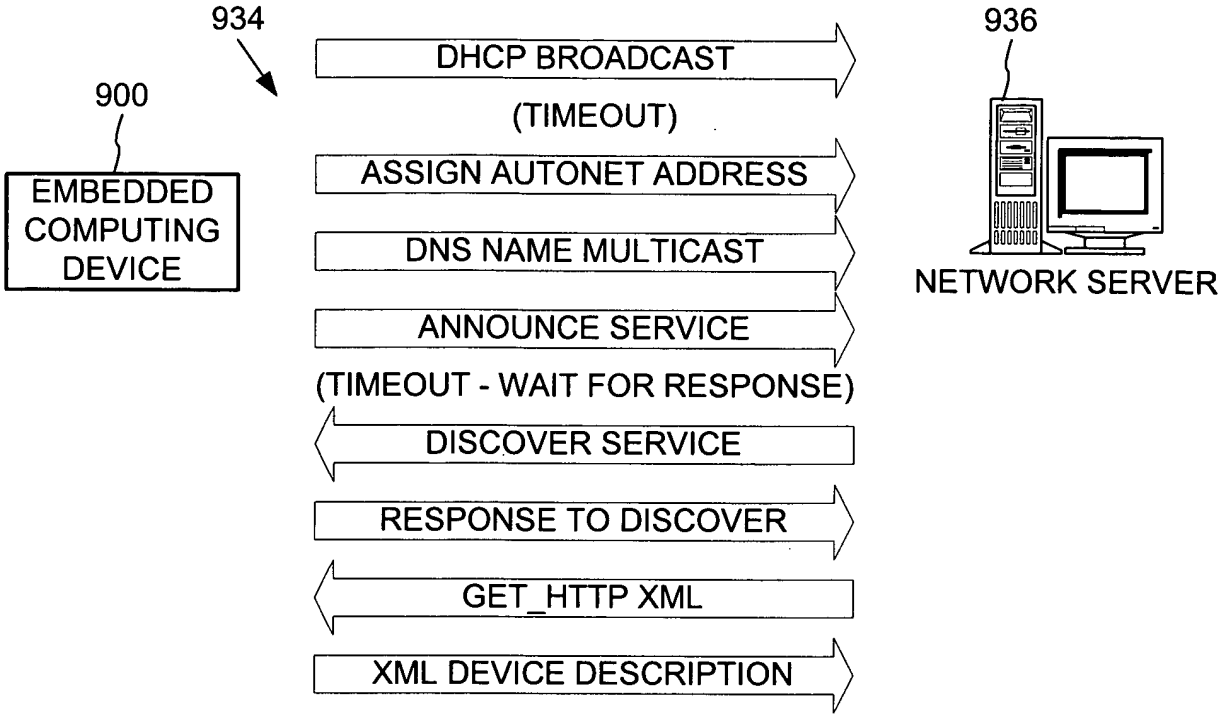


FIG. 25

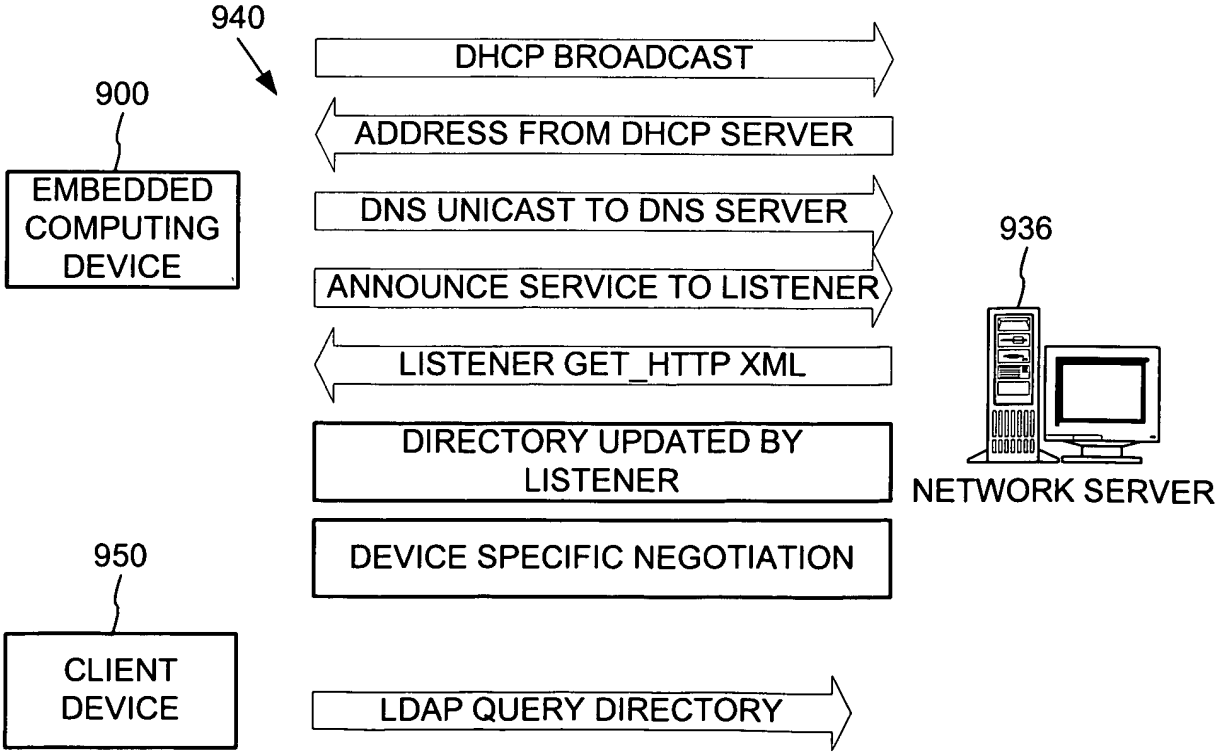

FIG. 26

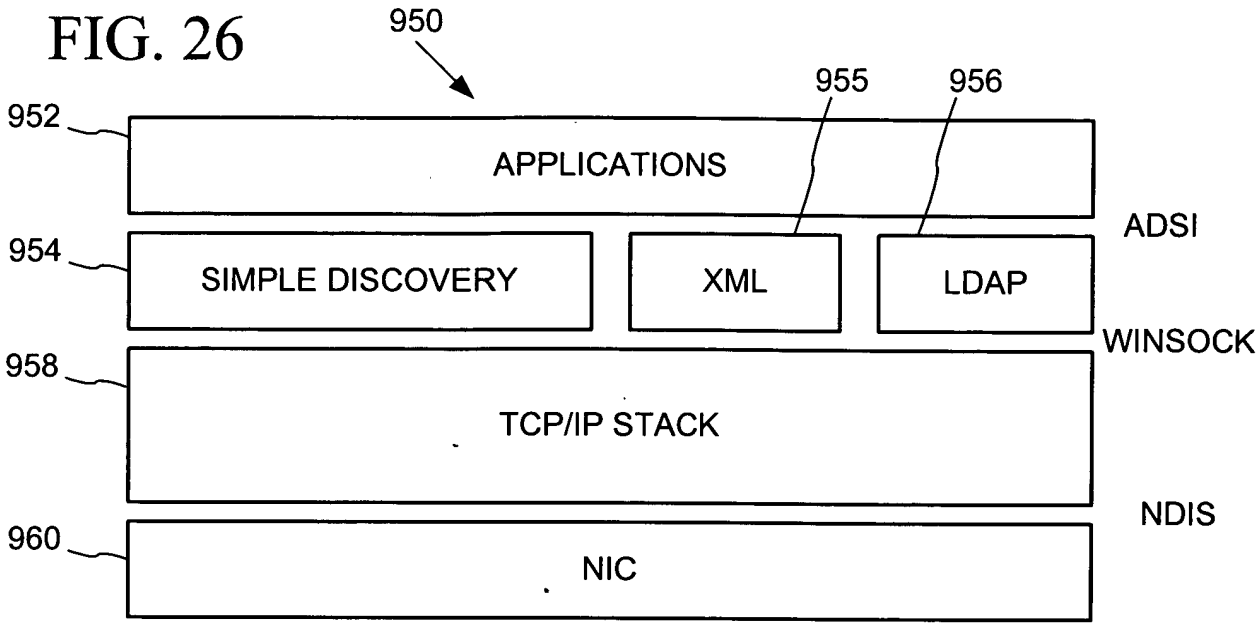

FIG. 27

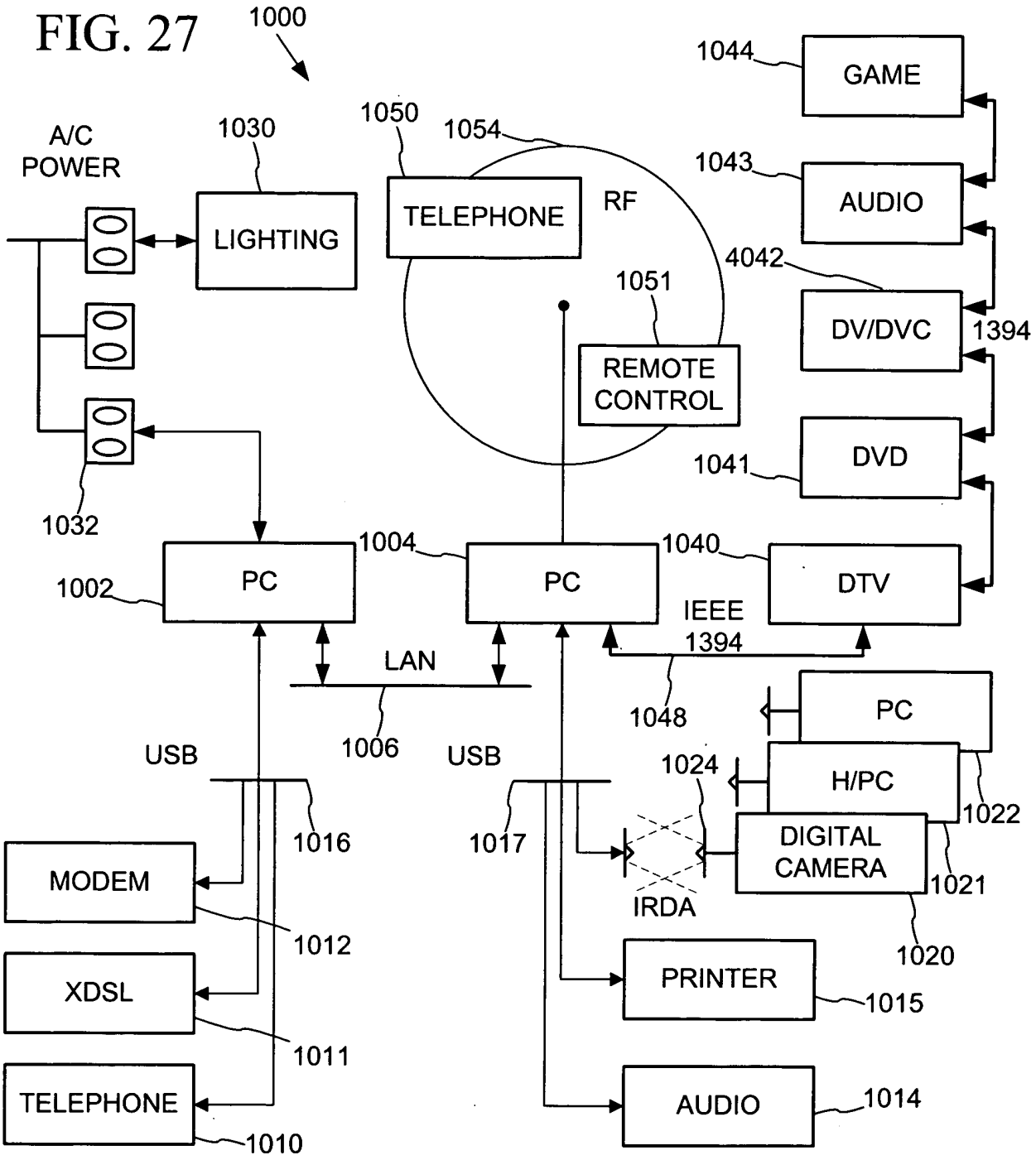
1000
1044
GAME
1050
1054
A/C
POWER
1030
1043
AUDIO
TELEPHONE
RF
LIGHTING
4042
1051
DV/DVC
1394
REMOTE
CONTROL
1041
DVD
1032
1004
1040
1002
PC
PC
DTV
IEEE
1394
LAN
1048
PC
1006
USB
USB
1024
H/PC
1022
1016
1017
DIGITAL
CAMERA
MODEM
IRDA
1021
1012
1020
XDSL
PRINTER
1015
1011
TELEPHONE
AUDIO
1014
1010

FIG. 28

```
[
object,
uuid(<foo>),
dual,
helpstring("IUPNPDevice interface"),
pointer_default(unique)
]
   interface IUPNPDevice : IDispatch
{

   [propget, id(DISPID_UPNPDEVICE_DESCRIPTIONDOCUMENT),
    helpstring("returns the document from which the properties of this device are
being read")]
   HRESULT DescriptionDocument([restricted, hidden, out, retval]
IUPNPDescriptionDocument ** ppuddDocument);
        purpose: returns the document from which the properties of this device are
being read.
        parameters: ppuddDocument, A reference to the description document
object from which data about the device is being read.  This must be freed when no
longer needed.
        return values: S_OK, ppuddDocument is a refernce to the device's
description document.

   [propget, id(DISPID_UPNPDEVICE_ISROOTDEVICE),
    helpstring("denotes whether the physical location information of this device can
be set")]
   HRESULT IsRootDevice([out, retval] VARIANT_BOOL * pvarb);
        parameters: pvarb, the address of a VARIANT_BOOL that will receive the
value of VARIANT_TRUE if the current device is the topmost device in the device
tree, and will receive the value of VARIANT_FALSE otherwise.
        return values: S_OK, varb is set to the appropriate value
        note: if a device is a root device, calls RootDevice() or ParentDevice() will
return NULL


   [propget, id(DISPID_UPNPDEVICE_ROOT),
    helpstring("returns the top device in the device tree")]
   HRESULT RootDevice([out, retval] IUPNPDevice ** ppudDeviceRoot);
        purpose: returns the top device in the device tree
...
```

FIG. 29

...

parameters: ppudDeviceRoot, On return, this refers to the "root" device of the current device tree. The root device is the topmost parent of the current device. If the current device is the root device this method will set *ppudDeviceRoot to null, and return S_FALSE.

return values: S_OK, *ppudDeviceRoot contains a reference to the root device. S_FALSE, the current device is the root device. *ppudDeviceRoot is null.

[propget, id(DISPID_UPNPDEVICE_PARENT),
helpstring("returns the parent of the current device")]
HRESULT ParentDevice([out, retval] IUPNPDevice ** ppudDeviceParent);

parameters: ppudDeviceParent, On return, if the device has a parent, this is the address of a IUPNPDevice object which can describe the parent. This must be released when no longer needed. If the device has no parent (it is a "root" device), than this value will be set to null.

return values: S_OK, ppudDeviceParent contains a reference to the device's parent. S_FALSE, the current device is the root device, which has no parent. *ppudDeviceRoot is null.

[propget, id(DISPID_UPNPDEVICE_CHILDREN),
helpstring("returns a collection of the children of the current device")]
HRESULT Children([out, retval] IUPNPDevices ** ppudChildren);

parameters: ppudChildren, On return, this is the address of a newly-created IUPNPDevices collection that can enumerate this device's children. This must be released when no longer needed. If the device has no children, this method will return a collection object with a length of zero.

return values: S_OK, ppudChildren contains a list of the device's children.

[propget, id(DISPID_UPNPDEVICE_UDN),
helpstring("returns the UDN of the device")]
HRESULT UniqueDeviceName([out, retval] BSTR * pbstrUDN);

parameters: pbstrUDN, On return, this contains the address of a newly-allocated string which contains the device's Unique Device Name (UDN). The UDN is globally unique across all devices - no two devices will ever have the same UDN. This value must be freed when no longer needed.

return values: S_OK pbstrUDN contains the UDN of the device

...

FIG. 30

```
...
 [propget, id(DISPID_UPNPDEVICE_DISPLAYNAME),
    helpstring("returns the (optional) display name of the device")]
   HRESULT DisplayName([out, retval] BSTR * pbstrDisplayName);
        parameters: pbstrDisplayName, On return, this contains the address of the
device's display name.  This value must be freed when no longer needed.  If the
device does not specify a display name, this parameter will be set to null.
        return values: S_OK, bstrDisplayName contains the display name of the
device.  pbstrDisplayName must be freed.  S_FALSE, the device did not specify a
display name.  *pbstrDisplayName is set to null.
        note: it is possible for multiple devices to have the same display name.
Applications should use UniqueDeviceName() to determine if two device objects
refer to the same device.

   [propget, id(DISPID_UPNPDEVICE_CANSETDISPLAYNAME),
    helpstring("denotes whether the physical location information of this device can
be set")]
   HRESULT CanSetDisplayName([out, retval] VARIANT_BOOL * pvarb);
        parameters: pvarb, the address of a VARIANT_BOOL.  This is true (!=0) on
return when the device's display name can be set (via SetDisplayName)
        return values: S_OK         varb is set to the appropriate value

   [id(DISPID_UPNPDEVICE_SETDISPLAYNAME),
    helpstring("sets the display name on the device")]
   HRESULT SetDisplayName([in] BSTR bstrDisplayName);
        parameters: bstrDisplayName, the value to set the device's display name to.
        return values: S_OK, varb is set to the appropriate value.
        note: On success, this method sets the display name used by a device.
Note that this method changes the display name on the device itself, not simply on
the local object.  This will block while the name is being set.
Additionally, this change will be made on the device alone, and will not be reflected
in the current device object.  After a successful call to this method, DisplayName
will continue to return the 'old' value).  To read the device's current name, the caller
must re-load the device's description.

   [propget, id(DISPID_UPNPDEVICE_DEVICETYPE),
...
```

FIG. 31

...
helpstring("returns the device type URI")]
HRESULT Type([out, retval] BSTR * pbstrType);
parameters: pbstrType, On return, this contains the address of a newly-allocated string containing the device's type URI. This value must be freed when no longer needed.
return values: S_OK, bstrType contains the type URI of the device, and must be freed when no longer needed.

[propget, id(DISPID_UPNPDEVICE_SERVICES),
helpstring("returns the collection of services exposed by the device")]
HRESULT Services([out, retval] IUPNPServices ** ppusServices);
parameters: ppusServices, On return, this is the address of a newly-created IUPNPServices collection that can enumerate the services exposed by the device. This must be released when no longer needed. If the device exposes no services, this method will return a collection object with a length of zero.
return values: S_OK, pusServices contains a list of the device's children.

[propget, id(DISPID_UPNPDEVICE_SERVICEIDENTIFIER),
helpstring("returns the (optional) service identifier of the device")]
HRESULT ServiceIdentifier([out, retval] BSTR * pbstrServiceID );
parameters: pbstrServiceID, On return, this contains the address of a newly-allocated string containing the contents of the device's ServiceIdentifier element, if the device specifies one. This value must be freed when no longer needed. If the device does not specify a ServiceIdentifier value, this parameter will be set to null.
return value: S_OK, bstrServiceID contains the service identifier of the device. pbstrServiceID must be freed. S_FALSE, the device did not specify a service identifier. *pbstrServiceID is set to null.
note having a ServiceIdentifier is mutually exclusive with having services. Any device will either have a list of services or a ServiceIdentifier, but not both.

[id(DISPID_UPNPDEVICEDESCRIPTION_LOADSMALLICON),
helpstring("loads a small (titlebar-sized) icon representing the device, encoded in the specified format")]
HRESULT LoadSmallIcon([in] BSTR bstrEncodingFormat,
[out, retval] BSTR * pbstrIconURL);
parameters:
...

FIG. 32

...
bstrEncodingFormat, A string containing the mime-type representing the desired encoding format of the icon. pbstrIconURL, On return, *pbstrIconURL contains a newly-allocated string representing the URL from which the icon can be loaded. This string must be freed when no longer needed.
return values: S_OK, *pbstrIconURL contains a reference to an icon, encoded in the desired encoding format.

[id(DISPID_UPNPDEVICEDESCRIPTION_LOADICON),
helpstring("loads a standard-sized icon representing the device, encoded in the specified format")]
HRESULT LoadIcon([in] BSTR bstrEncodingFormat,
[out, retval] BSTR * pbstrIconURL);
parameters: bstrEncodingFormat, A string containing the mime-type representing the desired encoding format of the icon. pbstrIconURL, On return, *pbstrIconURL contains a newly-allocated string representing the URL from which the icon can be loaded. This string must be freed when no longer needed.
return values: S_OK, *pbstrIconURL contains a reference to an icon, encoded in the desired encoding format.

[propget, id(DISPID_UPNPDEVICEDESCRIPTION_PRESENTATIONURL),
helpstring("obtains a presentation URL to a web page that can control the device")]
HRESULT PresentationURL([out, retval] BSTR * pbstrURL);
parameters: pbstrURL, on return, the address of a newly-allocated string containing the web-page-based control URL. If the device did not specify a presentation URL, an empy string ("") will be returned.
return values:S_OK, bstrURL contains a newly-allocated URL that must be freed when no longer needed. S_FALSE, the device does not have a presentation URL. pbstrURL is set to null.

[propget, id(DISPID_UPNPDEVICEDESCRIPTION_PHYSICALLOCATION),
helpstring("a set of properties describing the device's physical location")]
HRESULT PhysicalLocation([out, retval] IUPNPPropertyBag * pupl);
parameters: pupl on return, the address of a newly-allocated UPNPPropertyBag object which contains information about the device's physical location
return values
...

FIG. 33

...

S_OK upl contains a newly-allocated object that the caller must free when it is no longer needed.

note: if the object does not provide any description information, an empy property bag will be returned. See SetPhysicalLocation for a listing of defined values in a physical location property bag.

[propget,
id(DISPID_UPNPDEVICEDESCRIPTION_CANSETPHYSICALLOCATION),
helpstring("denotes whether the physical location information of this device can be set")]
HRESULT CanSetPhysicalLocation([out, retval] VARIANT_BOOL * pvarb);

parameters: pvarb the address of a VARIANT_BOOL. This is true (!=0) on return when the device's physical location can be set (via SetPhysicalLocation)

return values: S_OK varb is set to the appropriate value

[id(DISPID_UPNPDEVICEDESCRIPTION_SETPHYSICALLOCATION),
helpstring("writes a set of properties describing the device's physical location to the device")]
HRESULT SetPhysicalLocation([in] IUPNPPropertyBag * pupl);

parameters: pupl A UPNPPropertyBag object which contains the name-value pairs representing the device's current location. the function will not free the object.

return values: S_OK he device has been updated with the supplied physical location information

note: the following are standard values in the physical location property bag: country, campus, building, floor, wing, room, latitude, longitude, altitude. These values can be used programmatically to implement sorting or filtering functionality based on the device's location. Additionally the property bag supports the following value: description, which contains a user-displayable string representing a device's location which does not have programattic significance. Additionally, the physical location update will be made on the device alone, and will not be reflected in the current device object. After a successful call to this method, PhysicalLocation will continue to return the 'old' value. To read the device's current name, the caller must re-load the device's description.

}

...

FIG. 34

```
...
[propget, id(DISPID_UPNPDEVICEDESCRIPTION_PRODUCTNAME),
    helpstring("a displayable string containing the product name")]
   HRESULT ProductName([out, retval] BSTR * pbstr);
        parameters:  pbstr   on return, the address of a newly-allocated string
containing the product name of the device.
        return values:        S_OK pbstr contains a newly-allocated string that must
be freed when no longer needed.

   [propget, id(DISPID_UPNPDEVICEDESCRIPTION_DESCRIPTION),
    helpstring("displayable summary of the device's function")]
   HRESULT Description([out, retval] BSTR * pbstr);
        parameters:  pbstr   on return, the address of a newly-allocated string
containing a short description of the device meaningful to the user.
        return values:        S_OK pbstr contains a newly-allocated string that must
be freed when no longer needed.

   [propget, id(DISPID_UPNPDEVICEDESCRIPTION_MODELNAME),
    helpstring("displayable model name")]
   HRESULT ModelName([out, retval] BSTR * pbstr);
        parameters: pbstr    on return, the address of a newly-allocated string
containing the manufactuer's model name of the device.
        return values: S_OK        pbstr contains a newly-allocated string that must
be freed when no longer needed.

   [propget, id(DISPID_UPNPDEVICEDESCRIPTION_SERIALNUMBER),
    helpstring("displayable serial number")]
   HRESULT SerialNumber([out, retval] BSTR * pbstr);
        parameters: pbstr    on return, the address of a newly-allocated string
containing the manufacturer's serial number of the device.
        return values: S_OK        pbstr contains a newly-allocated string that must
be freed when no longer needed.
        note: a device's serial number is not guaranteed to be globally unique.  The
DeviceUniqueName should always be used to distinguish devices.

   [propget, id(DISPID_UPNPDEVICEDESCRIPTION_MANUFACTURERNAME),
    helpstring("displayable manufacturer name")]
   HRESULT ManufacturerName([out, retval] BSTR * pbstr);
        parameters
...
```

FIG. 35

...
pbstr, on return, the address of a newly-allocated string containing the name of the device's manufactuer.
return values: S_OK, pbstr contains a newly-allocated string that must be freed when no longer needed.

[propget, id(DISPID_UPNPDEVICEDESCRIPTION_MANUFACTURERURL),
helpstring("URL to the manufacturer's website")]
HRESULT ManufacturerURL([out, retval] BSTR * pbstr);
parameters: pbstr, on return, the address of a newly-allocated string containing the URL of the manufacturer's website.
return values: S_OK, pbstr contains a newly-allocated string that must be freed when no longer needed.

[propget, id(DISPID_UPNPDEVICEDESCRIPTION_MODELNAME),
helpstring("displayable model name")]
HRESULT ModelName([out, retval] BSTR * pbstr);
parameters: pbstr, on return, the address of a newly-allocated string containing the manufactuer's model name for the device.
return values: S_OK, pbstr contains a newly-allocated string that must be freed when no longer needed.

[propget, id(DISPID_UPNPDEVICEDESCRIPTION_SUPPORTLIST),
helpstring("technical support contact information")]
HRESULT SupportList([out, retval] BSTR * pbstr);
parameters: pbstr, on return, the address of a newly-allocated, multi-line string containing phone numbers and other information that can guide the user to technical support. This string must be freed when no longer needed.
return values: S_OK, pbstr contains a newly-allocated string that must be freed when no longer needed.

[propget, id(DISPID_UPNPDEVICEDESCRIPTION_FAQLIST),
helpstring("FAQ access display information")]
HRESULT FAQList([out, retval] BSTR * pbstr);
parameters: pbstr, on return, the address of a newly-allocated, multi-line string containing FAQ information that can provide the user with URLs at which device FAQs may be located.
return values: S_OK, pbstr contains a newly-allocated string that must be freed when no longer needed.
...

FIG. 36

```
...
 [propget, id(DISPID_UPNPDEVICEDESCRIPTION_UPDATELIST),
    helpstring("information explaining where the user can update the device's
firmware")]
   HRESULT UpdateList([out, retval] BSTR * pbstr);
        parameters: pbstr, on return, the address of a newly-allocated, multi-line
string containing information and URLs from which the user can download updates
for the device's firmware.
        return values: S_OK, pbstr contains a newly-allocated string that must be
freed when no longer needed.
```

FIG. 37

```
[
object,
uuid(FDBC0C73-BDA3-4C66-AC4F-F2D96FDAD68C),
dual,
helpstring("IUPNPDevices Interface"),
pointer_default(unique)
]
IUPNPPropertyBag
{

   [propget, id(DISPID_UPNP_PROPERTYBAG_READ),
    helpstring("reads a value from the property bag")]
   HRESULT Read([in] BSTR bstrName, [out, retval] VARIANT * pvarResult);
        parameters: bstrName, name of the property to read.  case is ignored.
pvarResultvalue of the property.  if the property doese not exist, this is of type
VT_EMPTY
        return values: S_OK, the value was found in the property bag, and returned
in pvarResult.  S_FALSE, there was no value with the given name in the property
bag.  *pvarResult is of type VT_EMPTY

   [propget, id(DISPID_UPNP_PROPERTYBAG_WRITE),
    helpstring("writes a value to the property bag")]
   HRESULT Write([in] BSTR bstrName, [in] VARIANT * pvarValue);
        parameters: bstrName, name of the property to write.  case is preserved
when writing.  The supplied value will replace any other values of the same name,
even if they differ in case.  pvarValue, value of the property to write.
        return values: S_OK, the value was written to the property bag, replacing the
value currently associated with this property, if it existed.

   [propget, id(DISPID_UPNP_PROPERTYBAG_DELETE),
    helpstring("removes a value from the property bag")]
   HRESULT Delete([in] BSTR bstrName);
        parameters: bstrName, name of the value to remove from the property gab.
case is ignored when finding a value to remove.
        return values: S_OK, the value has been removed from the property bag.
S_FALSE, the value was not found in the property bag.

};
```

FIG. 38

```
interface IUPnPService : IDispatch
{
   [id(DISPID_UPNPSERVICE_QUERYSTATEVARIABLE),
    helpstring("method QueryStateVariable")]
   HRESULT QueryStateVariable([in] BSTR bstrVariableName,
                              [out, retval] VARIANT *pValue);

   [id(DISPID_UPNPSERVICE_INVOKEACTION),
    helpstring("method InvokeAction")]
   HRESULT InvokeAction([in] BSTR bstrActionName,
                        [in] VARIANT vInActionArgs,
                        [in, out] VARIANT * pvOutActionArgs,
                        [out, retval] VARIANT *pvRetVal);

   [propget, id(DISPID_UPNPSERVICE_SERVICETYPEIDENTIFIER),
    helpstring("property ServiceTypeIdentifier")]
   HRESULT ServiceTypeIdentifier([out, retval] BSTR *pVal);

   [id(DISPID_UPNPSERVICE_ADDSTATECHANGECALLBACK),
    helpstring("method AddStateChangeCallback")]
   HRESULT AddCallback([in] IUnknown  * pUnkCallback);

   [propget, id(DISPID_UPNPSERVICE_SERVICEID),
    helpstring("property Id")]
   HRESULT Id([out, retval] BSTR *pbstrId);

   [propget, id(DISPID_UPNPSERVICE_LASTTRANSPORTSTATUS),
    helpstring("property LastTransportStatus")]
   HRESULT LastTransportStatus([out, retval] long * plValue);
};
```

FIG. 39

```
[
object,
uuid(FDBC0C73-BDA3-4C66-AC4F-F2D96FDAD68C),
dual,
helpstring("IUPNPDevices Interface"),
pointer_default(unique)
]
   interface IUPNPDevices : IDispatch
{
[propget, id(1), helpstring("property Count")]
HRESULT Count(
[out, retval] long *pVal
);

[propget, id(DISPID_NEWENUM), helpstring("property _NewEnum")]
HRESULT _NewEnum(
[out, retval] LPUNKNOWN *pVal
);

[propget, id(DISPID_VALUE), helpstring("property Item")]
HRESULT Item(
[in] long lIndex,
[out, retval] VARIANT *pVal
);
};
```

FIG. 40

```
[
object,
uuid(3F8C8E9E-9A7A-4DC8-BC41-FF31FA374956),
dual,
helpstring("IUPNPServices Interface"),
pointer_default(unique)
]
interface IUPNPServices : IDispatch
{
[propget, id(1), helpstring("property Count")]
HRESULT Count(
[out, retval] long *pVal
);

[propget, id(DISPID_NEWENUM), helpstring("property _NewEnum")]
HRESULT _NewEnum(
[out, retval] LPUNKNOWN *pVal
);

[propget, id(DISPID_VALUE), helpstring("property Item")]
HRESULT Item(
[in] long lIndex,
[out, retval] VARIANT *pVal
);
};
```